AN EDUCATION THEORY OF FAULT FOR AUTONOMOUS SYSTEMS

William D. Smart, Cindy M. Grimm, and Woodrow Hartzog

AN EDUCATION THEORY OF FAULT FOR AUTONOMOUS SYSTEMS

William D. Smart,[*] *Cindy M. Grimm,*[**] *and Woodrow Hartzog*[***]

Automated systems like self-driving cars and "smart" thermostats are a challenge for fault-based legal regimes like negligence because they have the potential to behave in unpredictable ways. How can people who build and deploy complex automated systems be said to be at fault when they could not have reasonably anticipated the behavior (and thus risk) of their tools?

Part of the problem is that the legal system has yet to settle on the language for identifying culpable behavior in the design and deployment for automated systems. In this article we offer an education theory of fault for autonomous systems–a new way to think about fault for all the relevant stakeholders who create and deploy "smart" technologies. We argue that the most important failures that lead autonomous systems to cause unpredictable harm are due to the lack of communication, clarity, and education between the procurer, developer, and users of these technologies.

In other words, while it is hard to exert meaningful control over automated systems to get them to act predictably, developers and procurers have great control over how much they test these tools and articulate their limits to all the other relevant parties. This makes testing and education one of the most legally relevant point of failures when automated systems harm people. By recognizing a responsibility to test and educate each other, foreseeable errors can be reduced, more accurate expectations can be set, and autonomous systems can be made more predictable and safer.

---

[*] Professor, Collaborative Robotics and Intelligent Systems Institute, Oregon State University.
[**] Professor, Collaborative Robotics and Intelligent Systems Institute, Oregon State University.
[***] Professor of Law and Computer Science, Northeastern University School of Law and Khoury College of Computer Sciences.

INTRODUCTION

In the American tort law system, the concept of "fault" is one principle for shifting the risk of loss in any given context.[1] In other words, parties that are "at fault" due to their culpable behavior should be held liable for the harms they caused rather than requiring the victim of the harm to bear the loss. The concept of fault in legal regimes is often premised on failing to reasonably protect against foreseeable harms.[2] Judges and lawmakers ask what control (allegedly) culpable actors have over the system and the risks associated with the exercise, or failure to exercise, that control. Complex autonomous systems like those in self-driving cars, "smart" thermostats, and autonomous warehouses present a difficult problem for fault-based legal regimes like torts for one simple reason: they have the potential to behave in unpredictable ways.[3]

How can people who build and deploy automated and intelligent systems be said to be at fault when they could not have reasonably anticipated the behavior (and thus risk) of an automated and intelligent system? As a very real case, the Tesla autopilot crash on May 7, 2016, that killed Joshua Brown was partly user fault (he was watching a movie) but also the fault of a sensor system that thought a truck was a cloud.[4] Perhaps the software and hardware designers should have included this test case, but how many more cases like this are there, and how would you enumerate them? Is it fair to blame just the creators when automated systems act in unpredictable ways?

This dilemma has tied lawmakers, judges, and academics in knots. Some scholars have suggested that the difficulty in assessing fault in the

[1] *See* Robert E. Keeton, *Conditional Fault in the Law of Torts*, 72 HARV. L. REV. 401, 401-02 (1959) ("[C]ourts should leave a loss where they find it unless good reason for shifting it appears. . . . In modern Anglo-American tort law, fault has been considered the one generally acceptable reason for such loss shifting. For more than a century, at least, fault has been the principal theme of tort law."); Brown v. Kendall, 60 Mass. 292, 298 (1850) (holding that defendants are liable for certain harms in tort only if they intended to cause the harm or if they are at fault in causing them).

[2] A related concept is that of "duty" towards others. RESTATEMENT (THIRD) OF TORTS: PHYSICAL & EMOTIONAL HARM § 7 (AM. L. INST. 2010) ("An actor ordinarily has a duty to exercise reasonable care when the actor's conduct creates a risk of physical harm."); Those who breach their duty to others in the language of the tort of negligence are said to be at fault. David G. Owen, *Philosophical Foundations of Fault in Tort Law*, *in* PHILOSOPHICAL FOUNDATIONS OF TORT LAW 201 (David G. Owen ed., 1995); RESTATEMENT (THIRD) OF TORTS: PHYSICAL & EMOTIONAL HARM § 3 (2010); *See also* O.W. HOLMES, THE COMMON LAW 77, 96 (1881).

[3] *See* Harry Surden & Mary-Anne Williams, *Technological Opacity, Predictability, and Self-Driving Cars*, 38 CARDOZO L. REV. 121, 125 (2016).

[4] Neal E. Boudette, *Tesla's Self-Driving System Cleared in Deadly Crash*, N.Y. TIMES (Jan. 19, 2017), https://www.nytimes.com/2017/01/19/business/tesla-model-s-autopilot-fatal-crash.html?_r=0.

design of automated, intelligent, and machine learning systems means a strict liability regime that holds producers liable for harm regardless of fault is preferable to an arbitrary or attenuated finding of fault.[5] We agree with many of the valid reasons why strict liability for harms caused by autonomous systems might be the best approach. However, fault-based approaches might be more attractive to courts, lawmakers, and industry if the right framework and language were used to establish a basis of fault in the creation, deployment, and use of autonomous systems.

We think part of the problem with our discussion of fault is that we have yet to settle on the best approach and language to use to specifically target culpable behavior in the design and deployment for automated systems. The purpose of this paper is to offer an additional structured and nuanced way of thinking about the duties and culpable behavior of all the relevant stakeholders in the creation and deployment of autonomous systems. We argue that some of the most articulable failures in the creation and deployment of unpredictable systems lie in the lack of communication, clarity, and education between the procurer, developer, and users of automated systems. In other words, while it is hard to exert meaningful "control" over automated systems to get them to act predictably, developers and procurers have great control over how much they *test* and *articulate the limits* of an automated technology to all the other relevant parties. This makes education through testing and teaching one of the most legally relevant point of failures when automated systems harm people.

As part of our proposed framework for identifying culpable behavior, we identify four specific and foreseeable education-failure points in the creation, deployment, and use of automated systems which contribute to harm caused by the unpredictability of autonomous systems. These failures are *Syntactic* (failure of sensors to identify objects and actions–i.e., the truck is a cloud), *Semantic* (failure to correctly translate human intent into algorithms, i.e., detect all objects you might run into), *Testing* (failure to test the system in expected scenarios - i.e., testing back-lit white objects) and *Warning* (failure to clearly articulate the limitations of the system–i.e., tell the user that the

[5] David Vladeck has made a compelling case for strict liability for parties that cause harm via automated vehicles, writing: "There are four strong policy reasons to establish a strict liability regime for this category of cases. First, providing redress for persons injured through no fault of their own is an important value in its own right. . . . Second, a strict liability regime is warranted because, in contrast to the injured party, the vehicle's creators are in a position to either absorb the costs, or through pricing decisions, to spread the burden of loss widely. . . . Third, a strict liability regime will spare all concerned the enormous transaction costs that would be expended if parties had to litigate liability issues involving driver-less cars where fault cannot be established. . . . And fourth, a predictable liability regime may better spur innovation than a less predictable system that depends on a quixotic search for, and then assignment of, fault." David C. Vladeck, *Machines Without Principals: Liability Rules and Artificial Intelligence*, 89 WASH. L. REV. 117, 146–47 (2014).

sensors are not sufficient to operate with the sun in certain positions or that an automated car cannot be predictably operated off-road).

To clarify, we are not arguing that our theory of culpability is preferable in all contexts. We reiterate our belief that arguments to hold manufacturers of automated systems strictly liable for harm they cause in certain contexts are compelling. Our goal for this paper is much more modest. We simply aim to introduce another possible way to think about the justifications for the law to shift liability for harms related to the creation and use of automated systems. In doing so, we aim to add to the developing body of literature in this field that includes theories of strict liability, res ipsa loquiter, common carrier liability, and proximity-driven liability, among others.[6]

This article proceeds in three parts. In Part I, we highlight the control and predictability gap in autonomous systems. This part provides a brief background into the process of designing and implementing autonomous systems and highlights the limits of how predictable (and controllable) these systems can be in practice. This part also highlights how the limits of predictability pose problems for traditional fault-based legal regimes like tort law.

In Part II, we introduce our theory of fault as a failure to properly test and educate other stakeholders on the goals, limitations, and foreseeable errors of an automated system. Specifically, we articulate four education-failure points where lack of clarification between the stakeholders and identification of foreseeable errors results in automated systems (potentially) acting in harmful and unpredictable ways. These four education-failure points also provide a pathway for the stakeholders

[6] *See* Bryant Walker Smith, *Proximity-Driven Liability*, 102 GEO. L. J. 1777, 1779 (2014) ("This Article argues that growing proximity could significantly expand sellers' point-of-sale and post-sale obligations toward people endangered by their products."). *See also* Andrew Selbst, *Negligence and AI's Human Users,* 100 B.U. L. REV. 1315 (2020); Rebecca Crootof, *The Internet of Torts: Expanding Civil Liability Standards to Address Corporate Remote Interference*, 69 DUKE L. J. 583-667 (2019); Ryan Calo, *Robotics and the Lessons of Cyberlaw*, 103 CALIF. L. REV. 513, 555 (2015) [hereinafter Calo, *Lessons of Cyberlaw*] ("There will be situations, particularly as emergent systems interact with one another, wherein otherwise useful technology will legitimately surprise all involved. Should these systems prove deeply useful to society, as many envision, some other formulation than foreseeability may be necessary to assess liability."); M. Ryan Calo, *Open Robotics*, 70 MD. L. REV. 571, 582-83 (2011) [hereinafter Calo, *Open Robotics*]; "Tort law is ordinarily unwilling to let people injured through no fault of their own bear costs imposed by others. So the question then becomes, 'Who pays?' The only feasible approach, it would seem, would be to infer a defect of some kind on the theory that the accident itself is proof of defect, even if there is compelling evidence that cuts against a defect theory. There is precedent for courts making such an inference, which is simply a restatement of res ipsa loquitor." Vladeck, *supra* note 5, at 128; Jack Boeglin, *The Costs of Self-Driving Cars: Reconciling Freedom and Privacy with Tort Liability in Autonomous Vehicle Regulation*, 17 YALE J. L. & TECH. 171, 175 (2015); Julie Goodrich, *Driving Miss Daisy: An Autonomous Chauffeur System*, 51 HOUS. L. REV. 265, 267 (2013).

to provide or demand evidence of due diligence at a more fine-grained level by explicitly elucidating the actions they have or should have taken to reduce potential harm.

The relevant stakeholders in the development and use of any automated system are *end-users* (the people using the technology), *procurers* or *merchants* (the people putting together application-specific systems for the end-users), and *developers* (the people developing the low-level technology used in the systems).

There are at least four different kinds of educational failures that create problems in autonomous systems: Syntactic, Semantic, Testing, and Warning. These are not independent because education is a two-way street: For example, Semantic failures have as their counterpart Warning failures.

1) Syntactic failures occur when developers fail to communicate to procurers the many different ways in which robotic systems might fail to identify real world objects. This error occurs because of a mismatch between the precision of artificial sensors and the robustness of human senses. Developers must identify such possible failures and communicate them to procurers in order to provide a full picture of the many different implementation problems.

2) Semantic failures occur when the human-articulated goals and intentions for autonomous systems are not translated correctly into software. Semantic failure can occur both between developers and procurers, and procurers and end-users (procurers or end-users incorrectly expressing their requirements to the developer and procurer, respectively). Similar to the Syntactic failure, it is largely the procurer (developer)'s responsibility to educate the end-user (procurer) on how they have translated the end-user (procurer)'s human-language statements into algorithms, and to clarify vocabulary usage.

3) Testing failures occur when a necessary syntactic or semantic test is simply missing from the test set. This is largely the developer's fault, but could also be the fault of the procurer for not fully articulating all of the desired use cases. Testing failures can also occur when the necessary syntactic or semantic tests are not conducted appropriately or are otherwise invalid.

4) Warning failures occur when users are not appropriately made aware of avoidable problems caused by the unpredictability of systems (the developer to the procurer or the procurer to the end-user). With respect to end-users, these failures are widely recognized in the law of product safety as a warning failure to be balanced with design defects. Warning failures are, in many ways, the inverse of semantic failures. Warning failures flow from developer to procurer to end-user, while semantic failures flow from procurer to developer.

Part III considers the impact of conceptualizing education-failures as culpable behavior. By articulating and isolating these educational failures, courts and lawmakers could better attribute responsibility among multiple actors working together to create complex and unpredictable autonomous systems. Such a framework could practically and legally allocate responsibility for testing, translation, and communication. It could also encourage more innovation, iteration, and

realistic design and implementation that better sets and syncs with user expectations about how robotic systems will operate.
Automated systems will probably always find new ways to surprise us. But by recognizing the relevant parties' responsibility to test and educate each other, foreseeable errors can be reduced, more accurate expectations can be set, and autonomous robots can be made more predictable and safer. We cannot "control" automated technologies in the traditional sense. But we can control how much we learn and teach each other about how these systems might work.

I. THE CONTROL AND PREDICTABILITY GAP IN AUTONOMOUS SYSTEMS

Robots are controlled by computer software, and computer software is deterministic. Presented with the same set of inputs, it will produce the same set of outputs every time. However, the software that controls robots is also complex. In particular, it often consists of several programs running in parallel, cooperatively controlling the robot. It is also driven by sensor data, allowing the robot to respond intelligently to a changing world. These two facts, along with our pre-conceived (and often wrong) human assumptions about what a robot should do, conspire to make most robot systems unpredictable when operating in the real world.

As a concrete example, consider the incident in early February, 2015 in South Korea where a robot vacuum cleaner accidentally drove over a sleeping woman's hair, sucking it into the cleaning mechanism, necessitating the intervention of a team of paramedics.[7] The robot in question is simple by the standards of autonomous systems. However, an interaction between the sensor data, the control program, and the assumptions of its victim led to unpredictable behavior and harm.

The robot presumably did not sense the woman's hair as an obstacle, so the control software did not avoid it. The woman may have assumed that the robot would avoid her as she slept on a mat on the floor, although this may not have been a case that the designers of the system actually thought about, since sleeping on mats on the floor is a culturally-specific act, more common in Asian countries than in Europe or the Americas. Even a simple robot, in a relatively simple situation, acted in a way that did not match a human's predictions, and a relatively significant harm was caused.

The unpredictability of complex automated systems is both one of its greatest virtues and one of its greatest obstacles to safe, sustainable adoption in society. Much of the point of automating some tasks is to see if automated systems can more efficiently or effectively complete work in new and previously unknown ways. Scholars and professionals use the

[7] Justin McCurry, *South Korean Woman's Hair 'Eaten' by Robot Vacuum Cleaner as She Slept*, GUARDIAN (Feb. 8, 2015, 11;53 PM), https://www.theguardian.com/world/2015/feb/09/south-korean-womans-hair-eaten-by-robot-vacuum-cleaner-as-she-slept.

term "emergence" to describe this new exceptional trait of robotics.[8] Ryan Calo wrote, "emergent behavior can lead to solutions no human would have come to on her own. Something approaching creativity can emerge from feeding data into a complex system and allowing it to iterate toward a semi-arbitrary goal."[9]

Unfortunately, the complexity of the decision-making process for robots and the vagaries of human behavior easily render the robot's behavior unpredictable outside of highly controlled settings, at which point the system can become dangerous. A self-driving wheelchair might interpret an empty space on a city street as an empty "sidewalk," and thus, more desirable to drive through than a crowded sidewalk, simply because no curb was sensed at a certain point in the road.

Automated cars must deal with changing environments.[10] Rain and snow can render some robot sensors largely useless. They must also deal with unexpected situations, such as a shopping cart rolling into the street. It is difficult to test all of the possible unexpected situations cars might encounter. Even humans have difficulty with novel situations. But humans' basic knowledge and understanding of physics serve as a reliable backup. Autonomous systems have yet to replicate this reliable background knowledge.

When autonomous systems behave in predictable ways for a large number of circumstances, industry can design safeguards for their use and deploy them more broadly. Warnings to users are more effective for autonomous systems because the warnings can be made more specific and reliable. If an automated car has been thoroughly tested on paved surfaces and only becomes unpredictable with grass or other unfamiliar terrain, then these cars can either be designed to stay on the pavement or, if the uncertainty and risk is small enough, can be designed to merely warn users that the car will not function properly unless it is on the pavement. Such specific warnings are preferable to broad disclaimers about the unpredictability of a robot's behaviors, which do not accurately guide users, and indeed, force users to accept large amounts of risk when using an automated technology.

The challenge for industry, then, is to make robots as predictable as possible while still preserving the usefulness of emergent behavior. How can developers control an autonomous system so that it "behaves" in expected ways? How can they discover what robots will do in a new situation?

The answer is, of course, that no one can make robots completely predictable. Developers can provide guidelines, examples, and test data that they hope will cause systems to behave in a desired (and predictable) way. The autonomous system will generalize the data developers give it. But roboticists do not directly control how complex robots will respond to

[8] *See* Calo, *Lessons of Cyberlaw*, *supra* note 6, at 539.

[9] *Id.*

[10] *See* Harry Surden & Mary Anne Williams, How Self-Driving Cars Work (May 25, 2016) (unpublished manuscript) (available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2784465).

new input. Predicting a robotic system's behavior is challenging for two reasons.

First, the complex functions and algorithms underlying the system do not always interpolate data in a predictable way. The field of control theory has spent decades developing mathematical techniques that guarantee that small changes in a robot's sensor readings will result in predictable changes in behavior, over a well-specified range of possible sensor readings.[11] However, today's robots are often complex enough that they go beyond traditional control theory, and rely on other, less predictable mathematical models. These models can tackle harder problems, but they lack the predictability offered by control theoretical approaches.

As an example, consider the thermostat in your house. If you put it up by one degree, then the heat will come on, and the temperature will ramp up to the desired setting. This is because the system controlled by the thermostat is well-modeled by traditional control theoretic techniques, and the response to turning the control is predictable. Turn it a little, and you get a small surge of heat. Turn it a lot, and you get a bigger surge. The mathematical relationship between the inputs (the setting) and the outputs (the temperature) is simple and well understood. If we were to use a more complex model, such as the currently popular deep convolutional neural network model, we lose this predictability. There is no way to predict the effects of a small change in the thermostat setting, unless we have seen that exact small change before.[12]

Second, and more subtly, autonomous systems operate on sensors, internal representations, and outputs that only approximately (if at all) match similar human semantic notions. For example, a human in a red shirt may be represented as a blob of red pixels of a certain size in an image, which could easily be confused with a picture of an apple of the same size. A naïve human would be utterly confused by why the autonomous system labeled an apple as a person in a red shirt, but it makes perfect sense to the autonomous system.

The sensors that robots use are, at their heart, devices for turning physical properties of the world (which we cannot directly measure) into electrical signals (which we can measure). This transduction always involves small errors, known as measurement errors, due to the physical processes involved. It also contains no semantics about the world; red photons are red photons, regardless of whether they came from a shirt or an apple. And in the end, everything comes out as just a number.

When humans recognize and categorize objects, they rely not only on the colors that they see, but on a vast store of semantic knowledge about what they might be looking at. Shirts are approximately the size of

[11] *See, e.g.*, NORMAN S. NISE, CONTROL SYSTEMS ENGINEERING (Dan Sayre et al. eds., 7th ed. 2015), or any other introductory control theory textbook.

[12] Technically, this phenomenon is related to the number of parameters in the mathematical model of the system. The fewer parameters there are, the easier it is to predict. For a thermostat, the system can be modeled with three parameters. A modern convolutional neural network model has, literally, millions of parameters.

a human torso, and often appear wrapped around people. They are made of cloth, which reflects light in a certain way. They have a typical shape, and are not rigid. Apples, on the other hand, are small, round, and often found on trees. The problem with object recognition and classification on a robot system is that to be as reliable as a human, to make the same kind of mistakes, and to be predictable, we would have to encode all of this prior knowledge about the world. This is simply impossible, partly because there is so much of it, partly because it is inconsistent (apples are small, but an apple-shaped hot air balloon is still an apple, sort of, but also simultaneously not an apple), and partly because it is tricky for us to even articulate the background knowledge when asked.

The problem is to reliably classify objects based only on low-level syntactic information about the world. Is that cluster of 216 pixels with red values in the range 234 to 245, green values in the range 32 to 234, and blue values in the range 126 to 201 an apple? Is it a shirt? Is it something that we have never seen before? The algorithms that robots use to recognize and classify objects make decisions based on these numbers, and perhaps other numbers too. However, these decisions are poor things, compared to the rich, nuanced way that humans carry out the same task. This distinction lies at the heart of the predictability problem. People and machines are using different sensors and radically different reasoning processes; no wonder the failures are hard to predict.

A. We Want Control, But the Best We Can Do is Approximate

The more control that roboticists attempt to assert over robots in particular circumstances, the less useful those robots become outside of those contexts. In other words, control comes at the expense of generalizability. What makes autonomous systems appealing is their ability to function under a wide range of circumstances, *sensing* their environment then *acting* based on that input. The actual control that developers have over systems comes down to creating and implementing guidelines and examples of desired behavior—"If you see something like Example A, then do Action B." For example, whenever a robot approaches a person, roboticists can program the robot to slow down or swerve aside.

Fundamentally, a developer's ability to control and predict the behavior of autonomous systems comes down to how thoroughly we have presented said system with examples of desired behavior, and how well we have matched our human, semantic notions of behavior with the corresponding capabilities of the autonomous system. As humans, we are fairly good at predicting what "expected" behavior is for other humans. For example, if you are waiting to turn onto a busy street and a driver slows down and flashes her lights at you, you would likely realize that this driver is going to allow you to turn into the street without hitting you. But even the most experienced developers can struggle with understanding how sensors, software, and learning combine to produce "sense" and "act" behavior in autonomous systems.

To compensate, developers perform educated "guessing" and as much testing as possible to create systems that approximate the desired behavior. For example, if the "human in the red shirt" detector uses a machine learning algorithm and initially classifies lots of red apples as shirts, the developer might present the system with lots of pictures of red apples and tag them as "not human" until the system learns the difference between the two. How many examples–and how much variation should be included in them (different lighting conditions, types of apples and shirts, different-sized people, backgrounds, sizes, orientation of the object in the image, etc. will dictate how good the system is at differentiating between the two. In the end, the designer can only provide so many examples. The better the examples, the better the autonomous system can approximate the desired behavior. But the resulting knowledge is still only an approximation. It will only be valid for inputs that are like (for a very complicated definition of "like") the images used to train the system. This approximation is about as close to "control" over a system as designers can get.

B. The Problem with Lack of Control and Unpredictability in Fault-Base Regimes

Complex automated systems pose unique problems to fault-based regimes because developers do not "control" robots quite the same way that car manufacturers "control" how airbags deploy, wheels and other parts work together, and steering wheels and brakes operate in their autos.[13] Whereas there often is a relatively traceable and predictable line between design and harm for many potentially harmful non-automated products like band saws (protective covers) and electrical kitchen appliances (short cords), existing software packages for object recognition and control systems are not as well understood and have far fewer default safety mechanisms built in.[14] This lack of safeguards for the

[13] *See generally* Soule v. Gen. Motors Corp., 882 P.2d 298, 308 (Cal. 1994); In re Toyota Motor Corp., 2013 WL 5763178, at *33 (C.D. Cal. Oct. 13, 2013); Potter v. Chi. Pneumatic Tool Co., 694 A.2d 1319, 1333 (Conn. 1997).

[14] Of course, this is not always true. This notion is complicated by the fact that the test for determining what constitutes a design defect is one of the most contested in all of tort law. *See* Vladeck, *supra* note 5, at 150 (citing Richard A. Epstein, *Products Liability: The Search for the Middle Ground*, 56 N.C. L. REV. 643, 647-49 (1978) (describing judicial confusion in assessing design defects); W. Page Keeton, *Product Liability–Design Hazards and the Meaning of Defect*, 10 CUMB. L. REV. 293, 298 n.23 (1979) ("The search for the universally acceptable definition of defect has been the most elusive one in the products liability field."); Joseph W. Little, *The Place of Consumer Expectations in Product Strict Liability Actions for Defectively Designed Products*, 61 TENN. L. REV. 1189, 1190 (1994) ("The difficult and politically contentious cases are those that involve allegations of defective design."); Marshall S. Shapo, *In Search of the Law of Products Liability: The ALI Restatement Project*, 48 VAND. L. REV. 631, 638 (1995) ("[A] crucial aspect of products liability law–perhaps the core concept, if any one idea may be described that way–lies in the definition of defect."); Marshall S. Shapo, *Products at the Millennium: Traversing a Transverse Section*, 53 S.C. L. REV.

technologies used in automated systems can be at least partially attributed to the general unpredictability of the system across broad contexts. How do you design protections for harms that are difficult to predict?

For example, many robots have a bump sensor that automatically stops the robot if it hits something. But this safeguard is less effective if the robot weights 300lbs and is moving at a good clip. "Visual" bump sensors (i.e., lasers) are more effective at detecting objects from a distance and slowing down in time. But laser bump systems have trouble with glass and thin items like table legs. Camera-based sensors might be able to find table legs but fail with people wearing plaid pants. The list goes on. There are so many different possible real-world variables, combined with sensor and control limitations that would cause the system to act in unpredictable ways, that all developers can really do is approximate and hope for the best. In this way, the act of determining the behavior of an automated system is more like coaching rather exerting direct control over its action.

Given this uncertainty, how should the law go about determining what constitutes unreasonably risky behavior with respect to the development and implementation of unpredictable robots? When a developer is not exactly sure how a robot will react, is any meaningful automation too risky? Or since there is little traditional control, is it unfair to attribute any culpability to the creation of unpredictable technologies?

The law could answer the question of when it is appropriate to hold developers and procurers responsible for harm in several different ways. It has developed, among other options, three different possible approaches as to when liability for harm should be shifted: when parties *intend* harm, when parties act *negligently* and are at fault for the harm that they cause, or when the parties should be held liable for all of the harm they cause regardless of whether they intended the harm or the foreseeability of the risks, commonly known as a "strict liability" approach.[15]

1031, 1033 (2002) ("However divided analysts of products law may be about definitions, most would agree that the heart of the matter in products liability is the concept of defect."); "The quest for understanding design defectiveness perennially vexes courts and accomplished products liability lawyers attempting to unravel design defect problems; delights law clerks, young associates, and law students, furnishing them with an occasion to display their erudition; and provides fertile grist for law professors aspiring for the renown that accompanies discovery of the key to any riddle wrapped in a mystery inside an enigma." David G. Owen, *Design Defects*, 73 MO. L. REV. 291, 291-93 (2008).

[15] *See generally* RESTATEMENT (THIRD) OF TORTS: PHYSICAL & EMOTIONAL HARM §1—3 (2010) (defining intentional harm, negligent harm, and strict liability.) *See also* RESTATEMENT (THIRD) OF TORTS: PROD. LIAB. § 2 (1998) (defining when a product is defective, including an intentional, negligent, and strict liability standard, and clarifying that strict liability holds a manufacturer liable even when all reasonable care was taken to prevent defectiveness.).

We do not yet wish to engage in the policy discussion over the preferable approach to shifting liability for harm caused by automated systems. Negligence and strict liability regimes both have likely costs and benefits that have yet to be fully realized in the contexts of robots. What we would like to do in this paper is suggest tools, frames, and a structured language for industry, courts, and law and policymakers in attributing culpability based on what developers and procurers have real, meaningful control over: discovering and conveying to everyone else in the process the knowns and the "known unknowns" of automated systems. Only the "unknown unknowns" are truly unforeseeable for those developing and procuring robots. In the next part, we offer up a theory of fault based on the failure to discover and communicate what we *do not* know about the behavior of automated systems.

II. FAULT AS FAILURE TO EDUCATE

In order to guard against unpredictability in automated systems, developers focus on rigorous testing under a variety of conditions. Testing is a well-understood concept in the software engineering community and is considered to be a vital part of the development process.[16] Most software takes some form of input data, transforms these data using some algorithm, and then outputs the result. For example, a robot might get input data from a digital camera (an image), apply a face-detection algorithm to it, and output the location of faces in the image. Testing involves taking known pairs of inputs and outputs and verifying that the software correctly performs the transformation. In the face-detection example, test data would consist of a set of images, with the locations of faces in them annotated. Often the tests are written before the software to be tested, a process known as *test-driven development*.

Testing can happen at a very low-level (face detector) or at a higher, more semantic level (identify when an unknown person enters the door after hours). Tests often build on each other, by first testing the low-level components, then combining the low-level tests into more system-level, complex behavior.

Testing robot software formally in this way yields three positive results: (1) it allows developers to say that, for the (hopefully representative) input data, that our software does the right thing; (2) it gives a characterization of the environments over which developers have tested and in which we have confidence that the system will behave as expected; and (3) it gives concrete evidence of due diligence in ensuring that the software operates as expected. If the testing data is sufficiently representative of the actual environments in which the robot will operate,

[16] *See generally* ROGER S. PRESSMAN, SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH 449 (Palgrave Macmillan, 7th ed., 2010). *See also* Cindy M. Grimm, William D. Smart & Woodrow Hartzog, *An Education Model of Reasonable and Good-Faith Efforts for Autonomous Systems*, PROC. 2018 AAAI/ACM CONF. ON AI, ETHICS, & SOC'Y 117 (Dec. 2018).

developers can claim that they have reasonably anticipated (and successfully dealt with) all foreseeable risks.

Testing and education accomplish two critical things for the design of automated technologies: (1) it gives developers the ability to make predictions about behavior, and (2) it gives developers the ability to know where these predictions are accurate (what we call the "limits" or "boundaries" of an automated technology). If a robot operates in environments that are well-represented by the testing data, then it should act predictably and thus give developers a reasonably accurate picture of foreseeable risks. However, if end users are clearly and reasonably warned against using a robot in an "off label" way in untested environments, fault for harm now more justifiably rests with the end-user or procurer instead of the developer, since they are using the robot in a way that it was not intended to be used (assuming the warning was sufficient to enable the user to reasonably avoid harm).

Explicit tests and mutual education of the parties could provide courts and lawmakers a specific and articulable kind of blameworthy conduct in the creation and use of an automated system. Focusing on education and testing as the locus of fault could also provide a more refined breakdown of who (end-user, procurer, or developer) is responsible for what tests (low-level component or higher, system-level tests). We now more formally break down testing failures into four categories: Syntactic, Semantic, Testing, and Warning.

A. Syntactic Failure

The most basic failure mode in a robot system is a failure to correctly identify properties of the world or objects in it. We call these failures syntactic failures, and they often occur because of the mismatch between the precision and inherent measurement errors of artificial sensors and the robustness of human senses. Sensors fail in ways that are often hard for humans to comprehend, without a deep technical understanding of the physical processes going on in the sensor. Developers also, in general, do not have sensors that directly sense the things they want robots to identify in the world. This means that developers have to have a sense of what kinds of sensors are capable and available, and then estimate the things procurers desire to sense based on these measurements. Consider a robot trying to find red balls. First, we must define what we mean by "red." Most humans with normal vision have an intuitive understanding of "red": that it comes in many shades, that pink is a sort of red, and that blue is not red. However, for a robot, we must define a closed set of pixel values, in the red, green, and blue channels that most digital cameras use, that delineate the idea of "red." A color that is just outside of this set is not red in the same way that blue is not red, at least to the robot. If we get the boundary of the set wrong, the resulting system will detect red objects as not red, or vice versa.

To make matters worse, sensors generally do not measure what we want them to measure. Pixels in an image generated by a camera are not really measuring the color of the object in the image. They are measuring the amount of light in ranges of wavelengths (corresponding to red, green, and blue) that is bouncing off the object and into the camera lens. A red ball illuminated with a white light might look the same as a white ball illuminated by a red light to the robot. However, only one of the balls is intrinsically red. In the end, we generally do not have sensors for the things we want to measure. We must measure the things that we can (number of photons[17] of a particular set of wavelengths hitting a piece of silicon), and then estimate the things that we are actually interested in (whether or not there is a red object in the camera image) from these measurements, and understanding of physics, and a set of assumptions. These assumptions are often left implicit, and this lack of explicit information is one of the shortcomings that we are trying to address with this paper.

In a more realistic example, consider face detection software, now commonly available on smartphones and digital cameras. This software does not actually detect faces. Instead, it detects groups of pixel values that are correlated with there being a face in the image. Most of a time, these collections of pixel values are actually generated by a face, and the software works as expected. However, sometimes they are not. Many face detectors will trigger with sketched outlines of faces drawn on a whiteboard. Most will identify a face on a poster as being a face, although it is really only the representation of a face. Although it may seem like we are splitting hairs here, if a robot is programmed to patrol an art museum at night, and to photograph anyone it encounters, it may spend a lot of time taking images of long-dead Dutch merchants and their wives. *Ceci n'est pas une visage*.

Since robots decide what to do on the basis of sensor readings, these syntactic errors can easily cause a robot system to fail. If asked to identify all people in red shirts, the robot might instead end up finding all people in white shirts standing under a red light, or people-shaped objects draped in red cloth, and not a person in a red shirt standing under a blue light.

While developers might not be able to eliminate these problems, they can certainly lessen them with well-defined sets of tests which, in computer science, would be called *unit tests*. These are small-scale tests used to verify the functionality of a small part of a computer program. For example, the face detector or the red shirt detector. To conduct these tests, developers could gather a representative set of sensor data, label it (annotating the location of the faces, to the areas we consider to be "red"), and then verifying that the software behaves as expected on these input data. This allows developers to say that, for environments that are similar to the ones in which we collected the testing data, an automated system will work as advertised. As we discussed above, this allows developers and procurers to both show a good-faith effort to ensure that

[17] In reality, we cannot even measure these photons directly. What we end up measuring is the electrical charge generated when these photons strike and interact with the camera's imaging surface. We can estimate the number of photons from an understanding of physics and the amount of charge that is generated.

the system works properly and to explicitly define the environments in which the robot should be used.

B. Semantic Failure

Above syntactic failures are what we will call *semantic failures.* These are failures to accurately translate human *intent* into software. These failures will generally be the result of taking loose, often colloquial human terms, loaded with nuance, bias, and a wealth of background knowledge about how the world works, and trying to translate them into concrete numbers and logic that can be used in the software. Consider the example of a building security robot, tasked with patrolling a warehouse, approaching and questioning anyone it sees after business hours. While patrolling is a clear instruction for a human security guard, there are a lot of implicit assumptions built into it. For example, if the robot sees a person through the window, walking down the street outside, it should probably not leave the building. Once it questions a person and determines that they are allowed to be in the building, it should probably not question them a second time if they meet again. When the robot approaches a person, how close should it come to them before it stops? Humans will naturally pick a socially and culturally appropriate distance, but we must pin this determination down to a specific number in software. The default social distance for robots in the United States might not be appropriate in, for example, Japan, leading to a different interaction between the robot and the human.

It is inevitable that semantic failures will happen when we try to translate human terms into robot control software, however much we try to minimize them. However, as with syntactic errors, we can address them with testing protocols. In this case the tests will need to involve higher-fidelity simulations or replay of previously recorded data from a similar robot platform. The tests, which are similar to *integration tests* in software engineering, will take these data, run them through the control software on the robot, and record the success or failure of the system.

As with syntactic tests, described above, our intent is not to eliminate semantic failure by testing all possible inputs to the robot's control software (which would be impossible). Rather, it is to show a good-faith effort on the part of the developers to ensure that semantic errors do not show up as the result of exposing the robot to a representative set of inputs.

As a concrete example of this, consider again the warehouse patrolling robot. Putting aside for now serious concerns about the deployment of automated systems in law enforcement generally, suppose that we have such a system shadowing a human security guard. The robot follows the guard around, recording sensor data as it goes. When the human guard sees a suspicious person, they hit a button on the robot, telling it that this is someone that should be approached, and then walk over to confront the intruder. Doing this for a few nights will result in a database of labeled sensor information that we can test our robot software against. Running these data through our algorithms, we can verify that the robot successfully identifies intruders and does not approach nonintruders, using the human guard's actions as the gold standard for what is "correct" behavior. For well-understood environments, we can imagine a high-fidelity simulation taking the place of job-shadowing a real human. This would also allow help properly temper optimism regarding the feasibility of automated systems in more unpredictable environments.

C. Testing Failure

A *testing failure* occurs when a necessary syntactic or semantic test is simply missing from the test set. This can occur both at the syntactic level when a specific condition is not tested for (a red shirt under a red light), or at the semantic level when sets of conditions do not adequately capture the human intent. This can happen because of missing groups of conditions or, more subtly, because the set of semantic tests do not take into account the differences between how humans perceive and label objects, and how robots do.

In the security guard example above, suppose all of the test data were captured on moonless Friday nights when only a small number of people were around. As humans, we can easily generalize this situation to a full moon on a Monday night, however a robot may not be able to. A full moon changes the colors of the images, confusing the people detector (syntactic test failure). Multiple people in the same space may confuse the person detector, making it difficult for the robot to detect individuals. Even if it succeeds in identifying all of the individuals, it may not know that it needs to approach each person individually and make sure it identifies all of them (semantic test failure). Or, the robot was also trained on groups of people, but only under moonless conditions–it fails under full moonlight conditions with multiple people (semantic test *sensor* failure).

A complete test set of all possible conditions is generally infeasible to enumerate, let alone produce. Usually, the test set is determined by balancing the cost of including that test (or risk of not including it) with the likelihood of that condition occurring.

D. Warning Failure

If there is any obligation to educate others with respect to automated systems that is already entrenched in the law, it is the obligation of manufacturers to accurately and clearly inform users about the limits of their automated technologies. In products liability law, this is known as a warnings failure.[18] A cognate area has to do with deceptive

[18] *See* RESTATEMENT (THIRD) OF TORTS: PROD. LIAB. § 2 (1998) ("A product is defective when, at the time of sale or distribution…is defective because of inadequate instructions or warnings. A product . . . (c) is defective because of inadequate instructions or warnings when the foreseeable risks of harm posed by the product could have been reduced or avoided by the provision of reasonable instructions or warnings by the seller or other distributor, or a predecessor in the commercial chain of

advertising, which in the United States is prohibited by, among other things, Section 5 of the Federal Trade Commission Act, which prohibits unfair and deceptive trade practices.[19]

A great example of where warnings are needed most is with automated cars. There are countless stories of people who (mistakenly) believed that the old-fashioned cruise control would correctly control the speed of their car even on a crowded road. Now we have cars that *do* actually adjust the speed if there is a car in front- misleading people into thinking that the car "knows" when to slow down and speed up. In reality, the car is simply using a time-of-flight sensor that senses how far away the object in front is and adjusts the speed accordingly. If this sensor fails (the object is too tall, it has a long, thin object sticking out the back, has glass on the back…) the automated speed control will fail. Unfortunately, the average human has no knowledge of how this sensor works and is unlikely to recognize when the sensor has failed and they should take control of the car's speed.

Right now, consumers have unrealistic expectations regarding what automated systems like driverless cars are capable of.[20] The U.S. National Highway and Traffic Safety Administration (NHSTA) recently adopted a new classification system developed by the Society of Automotive Engineers that includes six levels of autonomy.[21] Jonathan Ramsey described the levels as follows:

> Level 0 is intermittent warning systems like blind-spot detection; Level 1 encompasses features that can monitor the environment and alter steering or acceleration and braking, like parking assist, which only controls steering, or adaptive cruise control (ACC) that only adjusts speed; Level 2 includes systems that simultaneously steer and change speed, like ACC with lane-centering, and traffic jam assist that maintains space in traffic and navigates shallow bends. Under all of these level definitions, the driver is still charged with monitoring the environment. At Level 3, an "Automated Driving System" can take over all driving functions and total monitoring of the environment–the caveat being that the driver is expected to be ready to take over if the vehicle strays off-course, or finds itself in a situation it can't handle. According to the SAE paper, at Level 3 "an ADS is capable of continuing to perform the dynamic driving task for at least several seconds after providing the [driver] with a request to intervene." So, while the vehicle should be able to fully monitor its environment, the driver must also be ready to take over in an emergency.[22]

---

distribution, and the omission of the instructions or warnings renders the product not reasonably safe.").

[19] 15 U.S.C. § 45.

[20] Jonathan Ramsey, *The Way We Talk About Autonomy Is a Lie, and That's Dangerous*, THE DRIVE (Mar. 8, 2017), http://www.thedrive.com/tech/7324/the-way-we-talk-about-autonomy-is-a-lie-and-thats-dangerous.

[21] SAE INT'L, TAXONOMY AND DEFINITIONS FOR TERMS RELATED TO DRIVING AUTOMATION SYSTEMS FOR ON-ROAD MOTOR VEHICLES (2018), https://www.sae.org/standards/content/j3016_201806/.

[22] Ramsey, *supra* note 20.

Here is where the problem lies. It turns out that people are incredibly bad at taking over in emergency situations. While there are a few reasons for why we are not good at taking over for self-driving cars, mainly, we just are not expecting it. When a car partially "takes over," people have a tendency to relax because they do not anticipate the need to stay alert. Some have called this the "handoff" problem.[23] Ramsey wrote,

> A self-driving car operating in autonomous mode can create in drivers a false sense of security and a susceptibility to distraction, making an emergency disengagement more dangerous because the point at which the driver needs to retake control of the vehicle is also a point at which he is poorly equipped to do so. The specification says, in effect, "Your car will drive for you, but you need to watch it while it does, just in case." Studies have shown humans aren't good at continuous monitoring without personal involvement, and even Ford engineers tasked with monitoring the brand's self-driving cars from behind the wheel doze off at such a high rate that the company recently announced it would skip Level 3 [partial] autonomy altogether.[24]

Bryant Walker Smith has observed that automation levels are often presented to the user as promises by the automaker, i.e., "'[w]e are promising that our system will do this under these conditions.'"[25] Smith said it will be necessary for automakers to clearly communicate those capabilities and those limitations to the user, "'not just once, not just in an owner's manual, but in real time, as the systems are operating.'"[26]

We agree. Smith's comments highlight what we would refer to as a warning failure on the part of the producers. The obligation to meaningfully warn or train end-users is consistent with existing tort doctrine. Consider the cases where airplane manufacturers allegedly failed to provide adequate training to pilots in the safe use of their aircraft.[27] Put simply, procurers must adequately warn end-users about the limits of the automated system and how to reasonably avoid harm while using it. If such a warning is not possible, as might be the case with the "handoff" problem, then procurers must design the technology to be safe regardless of end-user behavior or go back to the drawing board to avoid being at fault.

---

[23] *See* AM. ASS'N FOR JUST., DRIVEN TO SAFETY: ROBOT CARS AND THE FUTURE OF LIABILITY (Feb. 2017), http://www.justice.org/sites/default/files/Driven%20to%20Safety%202017%20Online.pdf.

[24] Ramsey, *supra* note 20.

[25] *Id.*

[26] *Id.*

[27] *See, e.g.*, Glorvigen v. Cirrus Design Corp., 796 N.W.2d 541 (Minn. Ct. App. 2011) (considering but ultimately rejecting failure-to-train claim); Driver v. Burlington Aviation, Inc., 430 S.E.2d 476 (N.C. Ct. App. 1993); Berkebile v. Brantly Helicopter Corp., 311 A.2d 140, 142 (Pa. Super. Ct. 1973), *aff'd*, 337 A.2d 893 (Pa. 1975). *See also* Vladeck, *supra* note 5, at 150 (2014).

III. Possible Utility

We think that articulating fault in terms of testing and educational failures has at least three possible utilities. It can help make systems more predictable, it can help better set the expectations of the parties, and it can help courts, lawmakers, and industry (through private ordering), allocate financial responsibility for harms caused by robots.

A. More Predictable Systems

Once educated, parties who are working together to create and implement autonomous systems are then faced with the choice of continued testing and revision, or limited implementation according to predictability. Identifying failure to educate as blameworthy conduct will prioritize information exchange by the parties. By providing the incentive to educate others to eliminate uncertainty and ambiguity, parties will no longer be able to plausibly deny knowledge of a system's relative unpredictability. Instead, they must own that unpredictability by accommodating it with further testing, designing safeguards, or providing adequate warning.

The framework we are proposing in this paper can also provide a useful set of guidelines for addressing the educational gulf. Specifically, we look at the following educational responsibilities:

1) Developers must communicate potential syntactic failure to Procurers;

2) Procurers must be semantically clear to Developers when creating system specifications;

3) Developers must communicate potential semantic failure to Procurers;

4) Procurers must clearly communicate acceptable use-cases to End-users and warn them of dangers they can reasonably avoid. Let us expand upon this a little:

1) Developers are fundamentally the entities who best understand the limitations and capabilities of the sensors and control systems they are creating. Due diligence on their part constitutes communicating how nouns and verbs (concepts like "person" and "detect") are translated into sensor readings, thresholds, and controls. Developers must explain (as best as possible) the level of available accuracy and predictability. An example would be a developer explaining to a procurer, "you asked us to find people in red shirts, but we do not have a 'person in a red shirt detector,' we can simply find red-colored, people-shaped blobs–and it will fail if there is a blue light or an apple in range of the sensor."

2) Procurers must clearly communicate their requirements *in the language of the available sensors and as a set of clearly defined scenarios*. More specifically, they must identify expected conditions the sensor must operate under (moonlight, inside versus outside), or situations that might affect the robot's decision-making (lots of people versus solitary), and expected behavior (do not care about people outside the window). Recall the example of detecting red shirts from earlier in the paper. A shirt might appear red if it is, in fact, red. It might also appear red if it is

actually white and is seen under a red light. By making explicit the criteria for red (a specific set of sensor values that will map to the human concept of "red") and the set of environments under which this has been tested (white light, red light, blue light, etc.), we can place effective bounds on how much we can trust the classification of the sensor.

3) In return, Developers must clearly communicate to Procurers how their semantic requirements are mapped to algorithms and sensor thresholds (person outside the window involves first detecting the person, then determining that the person is located past the wall of the building–potential failures or unpredictable behavior may arise if the robot's localization relative to the map of the building is incorrect).

4) Finally, the Procurers must be able to communicate where and how their system is "safe" to use to a naïve End-user. For example, a Procurer might warn an End-user, "This robot cannot operate effectively outdoors" or "This automated car is not safe to use off-road." If warnings about the limits of a system cannot be clearly communicated and reasonably followed for safe use, then it may be better not to deploy the autonomous system at all or to design safeguards to prevent untested or unpredictable uses. For example, if an automated car cannot function predictably and safely in inclement weather, or if the "handoff" cannot be done in a reliably safe way, then Procurers might be at fault if they publicly deploy the car under those conditions. In all cases, Developers and Procurers must actually *do* the required tests, or demonstrate that the potential harm caused by that test failing is not substantial enough to warrant the cost of implementing that test.

By more clearly communicating capabilities and requirements *in terms of those capabilities*, we potentially increase the predictability of the autonomous system–or at the very least, determine when we should *not* use it. The design of the vehicles must then be adjusted to accommodate the many different known ways in which a machine might malfunction and crash, including scenarios where some other external factor causes sensors to be applied in unintended ways. In this way, conceptualizing fault in terms of testing and education is harmonious with the requirement that automobiles generally be "crashworthy," meaning a "manufacturer has a duty to design and manufacture its product so as reasonably to reduce the foreseeable harm that may occur in an accident brought about by causes other than a product defect."[28]

[28] RESTATEMENT (THIRD) OF TORTS: PROD. LIAB. § 16 (1998) cmt. A; DAVID G. OWEN, PRODUCTS LIABILITY LAW § 17.3, 1079 (1st ed. 2005); "It is well settled that the manufacturer of a defective product may be held liable for a portion of a plaintiff's injuries in cases where the product itself played no role in causing the plaintiff's initial accident. This now-familiar concept is known as the 'crashworthiness' or 'second collision' doctrine of product liability. Under this doctrine, the manufacturer may be held liable for a plaintiff's 'enhanced' or aggravated injuries-those injuries over and above the injuries the plaintiff would have sustained as a result of the initial accident absent the product defect."
Barry Levenstam & Daryl J. Lapp, *Plaintiff's Burden of Proving Enhanced Injury in Crashworthiness Cases: A Clash Worthy of Analysis*, 38 DEPAUL L. REV. 55 (1988).

B. More Accurate Expectations

One of the biggest problems with automated systems has to do with unrealistic expectations about what robots can actually do. This is partially fueled by science fiction and particularly fueled by our simple burning desire to automate certain burdensome tasks. It is common to dream of napping on the commute to and from work. The rush to innovate often encourages unjustified optimism and cut corners. By finding fault for failure to test and educate, a legal regime could incentivize the parties to communicate to each other the cold, hard truth about the limits of an automated system.

A healthy dose of reality might dampen people's spirits regarding technological progress, but it will have a few real benefits. First, if the system were implemented correctly, it would result in better technical literacies and make the development process more efficient by reducing miscommunication. Additionally, it will help improve End-users' mental models because Procurers will be tasked with making sure the limitations of a system are either clearly communicated and that it is reasonably easy for people to safely use the product. If effective communication and warning to the end-users is not feasible, then the product must simply be made safer.

C. A Way to Decide Who Pays

David Vladeck has accurately summed up one of the key problems in assigning liability for harms caused by autonomous systems like driverless cars. He noted that manufacturers are typically assigned liability for injury, though it could be assigned to "the operator, owner, the manufacturer, the programmers, the designers, or all of them…" Alternatively, the law could require driver-less cars to insure themselves as a "legal 'person.'"[29]

Vladeck argues that holding the manufacturer liable makes sense for most technologies because "the manufacturer sets the price for the vehicle, and so the manufacturer can build in an 'insurance premium' into the vehicle's sale price to offset expected liability costs."[30] However, he also notes that one problem with limiting liability for harms only to the manufacturer of automated systems like driverless cars is that "with driver-less cars, it may be that the most technologically complex parts–the automated driving systems, the radar and laser sensors that guide them, and the computers that make the decisions–are prone to undetectable failure. But those components may not be made by the manufacturer."[31]

The problem of assigning liability based upon fault is real and difficult because of the complexity of automated systems. Even when multiple parties are held liable, unless every party is pre-determined to

[29] Vladeck, *supra* note 5, at 147-48.

[30] *Id.*

[31] *Id.*

be equally liable, the parties might contest their relative degree of fault. Conceptualizing fault in terms of failure to test and educate might help in some contexts to more accurately resolve these disputes by pointing to specific and demonstrable failures–missing or inadequate tests and communication–that could have prevented the harm. This would help disperse liability among the parties engaged in the development of automated systems and provide incentives for each party in the chain to make their products safe and fully circumscribe the contexts in which an autonomous system is demonstrably predictable.

CONCLUSION

We offer this education-based theory of fault as a provocation–a suggestion that there might be another structured way to conceptualize obligations and culpability in the creation and use of autonomous systems. But the theory must be developed much more if it is to be useful. More research is required to determine which kinds of tests are the most significant, what constitutes "failure" in terms of testing and education, and how the testing can be effectively carried out in practice. The last of these points is particularly relevant to real robots operating in the real world, since most testing, in the software engineering sense, is done automatically inside of a computer. How can we effectively test control algorithms that operate on sensor data without actively gathering those data at the time of testing? If we pre-record them, how can we be sure that they are representative of the operating conditions where the robot will be? Must we physically travel to each location where the robot might be used, and collect data there? If we can simulate these environments, how can we validate the results from the simulator, and ensure that they carry across to the real world? Without good answers to these questions, we are really just kicking the can down the road.

Externally, more work is required to speculate how such a theory might be legally implemented and how this framework interacts with existing theories on fault and liability. For example, conceptualizing fault as failure to test and educate could be useful in comparative fault regimes (sometimes called comparative "responsibility" regimes), where courts and juries are asked to assess the relative fault (or responsibility) of the parties.[32] More research is also necessary to determine how this framework might usefully interact with traditional concepts in products

[32] *See* RESTATEMENT (THIRD) OF TORTS: APPORTIONMENT LIAB. § 8 (2000) ("Factors for assigning percentages of responsibility to each person whose legal responsibility has been established include (a) the nature of the person's risk-creating conduct, including any awareness or indifference with respect to the risks created by the conduct and any intent with respect to the harm created by the conduct; and (b) the strength of the causal connection between the person's risk-creating conduct and the harm."); *See generally* Gregory C. Sisk, *Comparative Fault and Common Sense*, 30 GONZ. L. REV. 29, 30-31 (1995) ("The adoption of tort reform constituted the next logical step in a principled progression . . . toward the principle of comparative fault among all who contributed to an injury.").

liability law such as "crashworthiness" and "reasonable alternative design."[33]

This research began as a quest to demystify the creation and operation of automated technologies and create a structured approach to assessing the true points of meaningful control in building and using robots. As these systems become more common in public spaces, we need to develop a language that will allow the various stakeholders to communicate about the predictability of the system and a way to think about where the boundaries of this predictability lie. By starting with what the parties can control, which is testing, ordering, and warning, then compelling an information exchange to further refine the design of technologies and make automated systems more generally predictable, courts, lawmakers, and industry experts might be better empowered to articulate where and why things went wrong and who should be held responsible. Automated systems may be a quagmire for traditional fault-based systems because of their unpredictability. But like so much in life, the best way to remedy a paucity of knowledge is usually a good education.

[33] *See* RESTATEMENT (THIRD) OF TORTS: PROD. LIAB. § 16 (1998); Toliver v. Gen. Motors Corp., 482 So. 2d 213, 218 (Miss. 1985) ("To prove his case [crashworthiness based on defective design], the plaintiff may introduce evidence of industry standards, to show deviation therefrom, or an alternate design, to show the feasibility thereof."); Huddell v. Levin, 537 F.2d 726, 737-38 (3d Cir. 1976) (applying New Jersey law); Crispin v. Volkswagenwerk AG, 591 A.2d 966, 976-78 (N.J. Super. Ct. App. Div. 1991), *cert. denied*, 599 A.2d 162 (1991) (given the tendency of the VW seat to yield upon impact, VW had a duty to warn about the necessity to wear seat belts to protect occupants from injury in the case of collision. Crispin is a crashworthiness case in which the failure-to-warn issue was presented as relevant to increased harm). *Contra, e.g.*, Rahmig v. Mosley Mach. Co., 412 N.W.2d 56, 81-82 (Neb. 1987) (plaintiff does not have to prove alternative safer design); Couch v. Mine Safety Appliances Co., 728 P.2d 585 (Wash. 1986).